Scripting language

From Wikipedia, the free encyclopedia

"Scripting" redirects here. For other uses, see script.

A **scripting language, script language** or **extension language**, is a programming language that controls a software application. "Scripts" are often treated as distinct from "programs", which execute independently from any other application. At the same time they are distinct from the core code of the application, which is usually written in a different language, and by being accessible to the end user they enable the behavior of the application to be adapted to the user's needs. Scripts are often, but not always, interpreted from the source code or "semi-compiled" to bytecode which is interpreted, unlike the applications they are associated with, which are traditionally compiled to native machine code for the system on which they run. Scripting languages are nearly always embedded in the application with which they are associated.

A scripting language for an operating system is called a shell script. Shell scripts have some affinities with application scripting languages but their history is distinct and their influence has been more widespread, and they will not be discussed in detail in this article.

The name "script" is derived from the written script of the performing arts, in which dialogue is set down to be spoken by human actors. Early script languages were often called *batch languages* or *job control languages*. Such early scripting languages were created to shorten the traditional edit-compile-link-run process.

Contents

Historical overview

The first interactive shells were developed in the 1960s to enable remote operation of the first time-sharing systems, and these generated a demand for scripting, to relieve the human operator of the tedium of re-entering sequences of commands at a computer terminal keyboard, so from there were developed simple macro commands, files containing sequences of commands, which eventually developed into shellscripts. In a parallel development, the larger and more complex applications developed embedded scripting facilities, at first very rudimentary, to facilitate batch mode operation where a human operator would not be present to guide the program. Thus part of the program was devoted to interpreting instructions written by the user in a (usually quite specialised) instruction language--a computer program within a computer program.

As scripting facilities developed they achieved Turing-completeness, which meant that they could be used to express any computer algorithm that could be expressed by any other programming language. The skills and techniques developed in application programming were brought to bear. The nascent scripting languages acquired variables, sequencing, iteration, decisions, procedures, and even modules. In time some embedded scripting languages became complex enough to be used to extend the application. Emacs, first developed in the late 1970s, is one classic example still in daily use. Most of the editing functionality of Emacs is written in its general scripting language, Emacs Lisp, around a core written in C.

Some mainstream languages, such as Lisp, adapted well to embedding within applications by providing high expressive power in a small footprint. Others, such as Tcl and Lua, were specifically designed as general purpose scripting languages that could be embedded in any application or used on their own. Other systems such as Visual Basic for Applications (VBA) provided strong integration with the automation facilities of an underlying system. Embedding of such general purpose scripting languages instead of developing a new language for each application also had obvious benefits, relieving the application developer of the need to code a language translator from scratch and allowing the user to apply skills learned elsewhere.

Although historically most scripting languages have been interpretive rather than compiled languages, applications requiring speed require fast extensions. QuakeC, developed for use in scripting the Quake game engine (mid-1990s), is an example of a scripting language that is compiled to bytecode which is interpreted by a bytecode interpreter or virtual machine. This isn't exactly new. In the seventies, a common system command and scripting language of DEC's RSTS/E operating system prior to version 9 of the O/S was Basic, compiled to bytecode.

Developed from the start to run in applets under the control of a web browser, Java provided the ultimate in multiplatform client-side scripting. Compiled to bytecode, it runs inside a Java Virtual Machine which provides security for the end user and abstracts the programming environment to simplify the programmer's task. Java has since found far more important uses in large-scale server-side scripting, however.

Early web servers provided primitive scripting facilities to generate non-static web content in response to an incoming query by launching a program or shellscript to handle each individual task (Common Gateway Interface), but the relatively high cost of launching a process for each query led to the development of server systems such as ASP, PHP, and JSP, which remained memory resident and came with their own scripting languages specialised for the purpose of serving web content. In using modules scripted in Java, JSP provides a highly sophisticated software engineering infrastructure to web-based scripting.

Some software incorporates several different scripting languages. Modern web browsers typically provide a language for writing extensions to the browser itself, and several standard embedded languages for controlling the browser, including ECMAScript (more commonly known as Javascript), CSS, and HTML.

Types of scripting languages

Job control languages and shells

A major class of scripting languages has grown out of the automation of job control, which relates to starting and controlling the behavior of system programs. Many of these languages' interpreters double as command line interfaces such as the Unix shell or the MS-DOS COMMAND.COM. Others, such as AppleScript, add scripting capability to computing environments lacking a command-line interface.

GUI Scripting

With the advent of Graphical user interfaces came a specialized kind of scripting language for controlling a computer. These languages interact with the same graphic windows, menus, buttons, and so on that a system generates. These languages are typically used to automate repetitive actions or configure a standard state. In principle they could be used to control any application running on a GUI-based computer; but, in practice, the support for such languages depend on the application and operating system. Such languages are also called "macros" when control is through simulated keypresses or mouse clicks.

Application-specific languages

Many large application programs include an idiomatic scripting language tailored to the needs of the application user. Likewise, many computer game systems use a custom scripting language to express the programmed actions of non-player characters and the game environment. Languages of this sort are designed for a single application; and, while they may superficially resemble a specific general-purpose language (e.g. QuakeC, modeled after C), they have custom features that distinguish them. Emacs Lisp, while a fully formed and capable dialect of Lisp, contains many special features that make it most useful for extending the editing functions of Emacs. An application-specific scripting language can be viewed as a domain-specific programming language specialized to a single application.

Web browsers

Web browsers are typically used to render HTML, but in time a host of special-purpose languages has developed to control their operation. These include ECMAScript, a very versatile procedural scripting language superficially resembling Java, Cascading style sheets, which enable style metadata to be abstracted from content, XML which can be used for content in conjunction with style metadata, as an alternative to HTML , and XSLT, a presentation language that transforms XML content into a new form. Techniques involving the combination of XML and Javascript scripting to improve the user's subjective impression of responsiveness have become significant enough to acquire a name: AJAX. The Document Object Model standard ensures that all browsers respond in a predictable manner to the same Javascript.

The Mozilla project has developed its own system for extending the user interface of the browser itself, called XUL.

Web servers

On the server side of the http link, application servers and other dynamic content servers such as Web content management systems provide content through a large variety of techniques and technologies typified by the scripting approach. Particularly prominent in this area are JSP, PHP, and ASP, but other developments such as Ruby on Rails have carved out a niche.

Text processing languages

The processing of text-based records is one of the oldest uses of scripting languages. Many, such as Unix's awk, sed, and grep were originally designed to aid programmers in automating tasks that involved Unix text-based configuration and log files. Of primary importance here is the regular expression, a language developed for the formal description of the lexical structure of text, and used by all of these tools.

Perl was originally designed to overcome the limitations of these tools, but has grown to be one of the most widespread general purpose scripting languages.

General-purpose dynamic languages

Some languages, such as Perl, began as scripting languages but were developed into programming languages suitable for broader purposes. Other similar languages -- frequently interpreted, memory-managed, or dynamic -- have been described as "scripting languages" for these similarities, even if they are more commonly used for applications programming. They are usually *not* called "scripting languages" by their own users.

Extension/embeddable languages

A number of languages have been designed for the purpose of replacing application-specific scripting languages by being embeddable in application programs. The application programmer (working in C or another systems language) includes "hooks" where the scripting language can control the application. These languages serve the same purpose as application-specific extension languages but with the advantage of allowing some transfer of skills from application to application. JavaScript began as and primarily still is a language for scripting inside web browsers; however, the standardization of the language as ECMAScript has made it popular as a general purpose embeddable language. In particular, the Mozilla implementation SpiderMonkey is embedded in several environments such as the Yahoo! Widget Engine. Other applications embedding ECMAScript implementations include the Adobe products Adobe Flash (ActionScript) and Adobe Acrobat (for scripting PDF files).

Tcl was created as an extension language but has come to be used more frequently as a general purpose language in roles similar to Python, Perl, and Ruby.

See also

- Domain-specific programming language
- List of programming languages
- Macro and preprocessor languages
- Ousterhout's dichotomy
- Shebang (Unix)
- Web template languages

External links

- A study of the Script-Oriented Programming (SOP) suitability of selected languages (http://merd.sourceforge.net/pixel/language-study/scripting-language/) – from The Scriptometer.
- A Slightly Skeptical View on Scripting Languages (http://www.softpanorama.org/Articles/a_slightly_skeptical_view_on_scripting_languages.shtml) by Dr. Nikolai Bezroukov
- Scripting: Higher Level Programming for the 21st Century (http://home.pacbell.net/ouster/scripting.html) by John K. Ousterhout
- Are Scripting Languages Any Good? A Validation of Perl, Python, Rexx, and Tcl against C, C++,

and Java (PDF) (http://page.mi.fu-berlin.de/~prechelt/Biblio/jccpprt2_advances2003.pdf) — 2003 study
- Scripting on the Java platform (http://www.javaworld.com/javaworld/jw-11-2007/jw-11-jsr223.html) - JavaWorld

Retrieved from "http://en.wikipedia.org/wiki/Scripting_language"
Categories: Scripting languages
